Western Kentucky University
TopSCHOLAR®

Student Organizations | WKU Archives Records

1990

UA66/3/3 Journal of the A. C. M.

WKU Association for Computing Machinery

Follow this and additional works at: http://digitalcommons.wku.edu/stu_org

 Part of the Computer Sciences Commons, Organizational Communication Commons, and the Sociology Commons

Recommended Citation
WKU Association for Computing Machinery, "UA66/3/3 Journal of the A. C. M." (1990). *Student Organizations*. Paper 291.
http://digitalcommons.wku.edu/stu_org/291

This Newsletter is brought to you for free and open access by TopSCHOLAR®. It has been accepted for inclusion in Student Organizations by an authorized administrator of TopSCHOLAR®. For more information, please contact topscholar@wku.edu.

4/90

Western Kentucky University
Association for Computing Machinery

First Annual
Journal of the A.C.M.

A.C.M. OFFICERS

	1989 - 1990	1990 - 1991
Chairman	Thomas Vaught	Jay Snider
Vice-Chairman	John Ternent	Max Stallings
Secretary/Treasurer	Debra Nash	Debra Nash
Faculty Advisor	Art Shindhelm	Tom Cheatham

Editor-in-Chief : John Ternent

WKU—Printing paid from state funds. KRS 57.375.

TABLE OF CONTENTS

From The Editor's Desk

John Ternent

When Dr. Shindhelm asked if I was willing to prepare a newsletter for ACM, I thought it would be a pretty neat thing to do. Little did I know how much trouble this project would be. First of all, I figured that I really didn't care if my contributors gave me their articles on disk or as hard copy. I quickly learned the same lesson Dr. Cheatham learned with the alumni newsletter: always ask for the articles on diskette!!! Typing in Dave Beckley's article on fiber optics just about fried my brain.

Due to certain circumstances, such as a certain Research Methods class that some really stupid CS majors take, this newsletter did not come out when I expected it to. Therefore, some of the articles, such as the one on the microcomputer raffle, may be outdated. Others may be just plain wrong, as I am also the official proofreader for this project. Any errors in these articles are mine and mine alone.

Many thanks to our contributors: Chris, Dave, Dr. Cheatham, Dr. Sloan, Dr. Baur, Dr. Modesitt, Mrs. Wilson, and Dr. Shindhelm (even though he had to contribute something, since he is our illustrious leader). Many thanks to the guy who contributed the Weekly Smile. I told him I wouldn't use his name. Are you happy now, Todd? Oops. Sorry about that. Ha, ha.

Anyways, I hope that this newsletter becomes a tradition for our ACM chapter. Even though it is a lot of work, it is really a worthwhile project.

The VAX with Two Brains
by chris speth

Conversion of the heart, soul, and mind

Life's hard and then you get a new computer system. Recently, the student operators that work at Science and Tech Room 110 (the things some people will do for a T-shirt) and I converted all the functionality of the VAX 11/785 to the VAX 6320 which was purchased in October. The VAX 6320 is at least five times the speed and twice as much trouble as the VAX 11/785. The conversion was initially quite simple. The VAX 11/785 was taken down, a new disk drive was plugged into the 11/785, and a copy of the existing system disk was copied onto it. The new system disk was plugged into the VAX 6320 which subsequently booted. Imagine, putting the brains of a cat into a jaguar and then the jaguar being able to run from the neighborhood dogs.

Well, we had to tell the VAX 6320 to take a look in the mirror. Its first identity crisis was to notice that it was not a single processor but a tightly-coupled dual processor. It observed that it did not have 16 one-megabyte boards of memory resident, but rather 64 megabytes of memory on two boards. In addition, there was not a separate 16-bit microprocessor used to load the micro-code at boot time, but rather an eeprom boot chip resident on each of two CPU boards. The four boards of floating point processor were also gone, and hence incorporated on each of the CPU boards. Communication to the outside world would no longer be accomplished through 108 tunnels of 25 lanes each but down one access road 15 lanes wide.

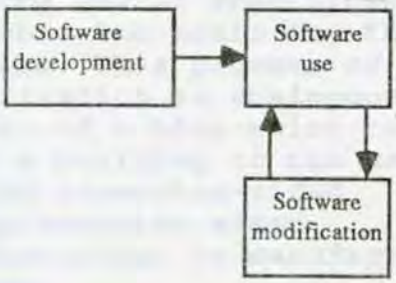
That proved to by my two-week migraine/Excedrin headache number 97. Understanding Ethernet involved devouring two 2 1/2 inch thick manuals in a weekend and bits and pieces of four others. This was a humbling experience. At first I was bound and determined to overcome this foreign architecture without any assistance. After a week of bouncing off various walls, operators, and manuals, I called DEC Technical Support. Within 15 minutes they talked me through the solution to my common mistakes and referred me to two other manuals.

The other disk drives and tape drive that were connected to the VAX 11/785 were disconnected and plugged into the VAX 6320. Finally, the cables that connected the VAX 11/785 to the campus-wide area network were plugged into the two DecServer 550's. The only difference the users should notice is a quicker response.

Other projects coming down the pike are the re-installation of the VAX 11/785 in the Nuclear Physics Lab to facilitate real-time data acquisition and setting up a local area network with the VAX 6320 as the server. I'm going to switch to Tylenol, cyanide version.

The Software Life Cycle
By Dr. Greg Baur

To understand the importance of software engineering, it is helpful to understand the software life cycle, which is concerned with what happens to a piece of software from the time the idea is born until the time it is discarded. The software life cycle is illustrated in Figure 1.



The figure shows that once the software has been developed, it enters a looping pattern between use and modification. The software stays in this loop until it is time to be discarded. Such a looping pattern is common for any product that is manufactured. The product cycles between use and repair/maintenance until it eventually wears out and is replaced.

Software does not wear out. After its development, it moves into the use phase where it is likely that someone determines there are errors in the software or that changes are required. The software then loops through this pattern because, once errors are corrected or changes are made, new problems may arise during the next use phase. A potential difficulty with making changes to software is the risk of causing new problems. For example, changes in an accounting procedure will cause changes to be made in a program that handles accounts payable or accounts receivable. It is often the case that these changes may cause problems that will not be discovered until some later time.

It is useful to examine at this point, the relationship between the development and modification phases of the software life cycle. The amount of time and effort spent in the modification phase is likely to be inversely proportional to the time and effort spent in the development phase. That is, more time and effort spent in the development phase may cause less time and effort to be required in the modification phase. If care is taken early-on to ensure proper system design and implementation, then the resulting system is more likely to meet most of the user's needs. Also, it is likely that there will be fewer errors to correct in the modification phase.

It is clear that a great deal of importance must be placed on the development phase. This is true in any software development situation, but is especially true with large software systems where many people are involved. A methodical approach using software engineering techniques is needed.

In a systematic approach, there are several required steps. Our discussion here will be brief; therefore, if you want further detail, you may wish to consult books such as Software Engineering: A Practical Approach by O. Pressman [McGraw-Hill, 1982] or Introduction to Systems Analysis and Design by J. Kendall [Allyn and Bacon, 1987].

The first step in the

velopment phase is to rmalize the problem ecifications.

It is at this point when he decision is made that ftware will be appropriate r the given application. The quirements are then isolated d production begins. These quirements are refined and come the specifications of he application. For kample, the application might equire that data be accessed nly by authorized personnel.

The set of specifications hus form the basis for the ftware. This process of rmalization is analogous to he use of a blue-print to uild a building or the use of AD/CAM (computer-aided esign/computer-aided anufacturing) to manufacture a roduct.

The formalization process hould maximize the nderstanding of the objective f the application by both the ser(s) and the software eveloper. If the process does ot work in this way, then dditional time and effort that an be avoided, will be eeded later in development to ake changes.

Formalization is probably he most difficult part of the evelopment phase. It requires he software developer (who may r may not do the actual rogramming) to be a skilled ommunicator to ensure clearly tated requirements. Often, sers are not good at rticulating their application eeds. Sometimes a manager ill provide limited input to ndicate, the parameters of the pplication. Except a short ime constraint for completion here may be little direction. he software developer s then responsible for most of he formalization and resenting the specifications to management for review.

The second step involves the design of the software system structure. At this point, the application must be divided into a series of tasks or modules. At the same time, techniques for interfacing the modules must be considered to minimize the time spent in future system maintenance. The software developer must ensure that all of the specifications have been accounted for in the design of the software system.

Implementation of the software system is the third step. This involves the actual creation of the software program. As in the design step, implementation is accomplished on a modular basis using the modules defined in the design process. As modules are completed, their interface to other modules in the software system must be thoroughly tested. This testing is tedious and at times is considered to be of little importance by system developers. Slighting the testing process in this step however, usually leads to the creation of larger and more difficult problems when the entire system is put together, which otherwise may have been avoided.

The fourth step involves the testing and debugging of the entire software system. Do all the modules interact properly with each other? What bugs remain that were not found in the design step?

Also at this time the overall software system is assessed. Does the software have satisfactory performance for the user and have all original specifications been met? These questions must be resolved now.

When this step of the software system development phase is completed, the system

Concluded on p. 13

LOCAL AREA NETWORKS
by Jay Sloan, A.C.R.S.

Students working on microcomputers in the laboratories at TCCW, Grise, and STH have been exposed to Academic Computing's Local Area Networks (LANs). We welcome this opportunity to provide users with information about the LANs, why they were installed, and what we expect to happen in the future.

Each of the LANs is based upon an 80386 machine running NOVELL's NetWare operating system. At present, the largest of the LANs is that in Grise Hall with a total of 77 machines connected in the second and third floor laboratories. Individual micros are linked to the servers by thinwire Ethernet interface cards. These cards are equipped with ROM chips which boot the LAN connection automatically upon powering up (though this may be sidestepped when stand-alone operation is desired).

The basic function of the LAN is to allow each of the lab's micros to access the server's hard disk as if it were installed in the individual machine. This function provides the rationale for the installation of the networks: software sharing. ACRS has agreed to supply a suite of general purpose software packages in each of its microcomputer laboratories; word processing (both WordStar and WordPerfect), a spread sheet (Lotus 1-2-3), and data base management (dBase IV). In addition, BASIC and SPSS-PC, a statistical analysis package, are also on the server menu. Beyond this standard set of resources, faculty may have other software and files placed on the server for their students' use.

This arrangement provides benefits to both the users and Academic Computing. From the user perspective, operations are cleaner and faster. Applications load much faster from the server than from diskette drives, not to mention the time saved in checking software in and out or dealing with defective disks. These same benefits apply to Academic Computing; others include facilitating updates, providing controls necessary to meet software licensing obligations, and saving a considerable amount over what it would have cost to equip a large fraction of the machines with individual hard disks.

Further, there is the general issue of conforming to the world beyond the campus' boundaries. Establishing the LAN environment is important since this is becoming typical of the work place. LANs are only a piece of the general connectivity puzzle, however. Our aims include connecting the LANs to one another as well as to the VAX and IBM systems, adding an existing lab in Cherry Hall to the STH LAN, and reaching into faculty offices.

A number of alternative paths will be explored. This Spring, another Ethernet LAN will be installed in STH using the VAX 6320 as a server. We expect to learn a good deal about the relative merits and costs of micro- versus minicomputer-based LANs. Finally, Ethernet may not be the best communications approach. Results from testing laboratories are certainly important, but our choices should be conditioned by comparative local

Concluded on p. 13

Speakers For ACM Meetings
by Art Shindhelm

Each year the ACM Faculty Advisor tries to line up a wide variety of speakers and films for the meetings that year. There is an official ACM Speakers Bureau intended to act as a pool of potential speakers to choose from. Last year we had one such speaker, Robert J. Tufts speak on "What's Wrong With Fourth Generation Languages". Unfortunately, it is usually necessary to arrange a tour of at least three schools for them to speak at in order for them to come. We also solicit speakers from local talent, such as faculty, staff and alumni. Each year we try to devote one meeting to job opportunities for students, including full time jobs as well as coop positions.

This year, as a result of hearing him at a conference I attended in Birmingham Alabama, we had Dr. Robert Hyatt, one of the writers of the Cray Blitz program. The Cray Blitz program was the 1987 world champion chess program.

If any student has a particular computer related topic or person that they would like to have a meeting devoted to, they should contact either the ACM faculty advisor or any ACM officer. After all, the club is meant to be for YOU!

UNIX
By Tom Cheatham

The UNIX Operating System was born in 1969 at AT&T Bell Labs. It was designed by programmers for programmers. In 1990 UNIX runs on more different platforms than any other operating system in the history of computing. It is available on practically every multi-user system in the world including IBM, DEC, Burrows, TI, etc, etc ... Why?

Several reasons: (1) It is written, except for maybe 10% of the very low-level code, in the C-language. Being written mostly in a high-level language makes it more easily ported to new platforms. (2) UNIX pioneered several powerful ideas that allow programs to communicate with each other: piping, redirection of IO, background processes, shell programming language, and others. (3) UNIX is an open system -- since 1974 Bell Labs has been licensing the source code for UNIX to universities for research and experimentation. We have all the source code for our Berkeley 2.10.1 UNIX for the PDP 11/44. I consider the source code my best tool in the graduate UNIX OS class. We also offer biterms that allow students to get acquainted with UNIX: CS245 C, CS 245 C-Shell, CS245 C++, CS245 PROLOG, and CS245 LISP. It's a good career move to gain some experience on this widely used Operating System. See you in one of my classes.

COMPUTER-BASED LEARNING, EXPERT SYSTEMS AND SOFTWARE ENGINEERING: ADVANCED TOOLS FOR ENGINEERING EDUCATION NOW AND IN 2001.

Dr. Kenneth L. Modesitt
Head, Department of Computer Science
Western Kentucky University
Bowling Green, KY 42101 (USA)
(502) 745-4642

English Abstract

ill design and build space station
om of today and the lunar and Martian
s and bases of tomorrow? Several
ced tools can address the shortage of
fied future engineers. Expert
ter systems permit experts to
mit their expertise to successors.
ter-based learning tools help people
difficult concepts by interactive
gues. Software engineering methods
e large programs to be built with a
r degree of reliability and safety.

German Abstract

Wer wird heute die Raumstation "Freedom" und in der Zukunft die Mond- und Marsstation entwerfen und bauen? Es gibt mehrere fortgeschrittene Möglichkeiten. Expert Computer Systems erlauben Experten, ihre Erfahrungen an Nachfolger weiterzureichen. Intelligente Lernsysteme helfen Studenten, schwierige Konzepte im gegenseitigen Dialog zu erlernen. Techniken in "Engineering Software" machen es möglich, grosse Programme mit einem höheren Grad von Zuverlässigkeit zu entwickeln.

What is the Problem?

he world has never lacked for
ems or "opportunities". Moreover, it
lso been recognized by at least one
rity that problems are not usually
d by the people who create them:
e world that we have made as a result
the level of thinking we have done
s far creates problems we cannot
ve at the same level of thinking at
ch we created them."
lbert Einstein
es not take a person of Albert's
lect to perceive that the world is
te with a multitude of significant
ems of our own making: the greenhouse
t, acid rain, de-forestation,
tion, bloody altercations over
aries of nations, crime, food
ies to needy areas, ad infinitum.
ome reasons for these include: lack
ney, lack of motivation, lack of
n, fear of loosing current resources,
The single most important lack
-wide, however, is that of
ically competent professional
iduals who can attack the gaps
en "what is" and "what should be."
the theme of this conference:
eering education 2000, as held in an
national environment.
s a specific example, figure 1 shows
earth of technical currently-
fied staff at a major aerospace
ration. The figure demonstrates the
h of senior knowledge, promise in be-
ng technical staff, and a gap in the
ctive mid-range. Similar plots would
nationwide for most engineering
ed companies and universities!

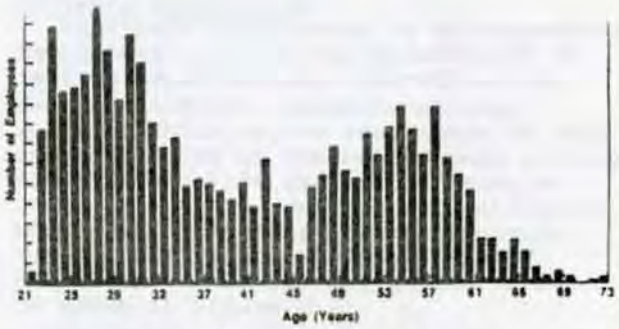


Figure 1. Distribution of Engineers

Solving the Problem

There are a number of approaches to solve the problem of increasing the effectiveness of a resource, i.e., the pool of technically qualified people.

1. Find a new supply of the resource.
   Increase enrollment efforts for women, minorities, third world, under-represented populations.
   Retrain other professionals.
2. Find a new type of resource.
   Use robots to perform tasks now done by clever humans. Not viable!
3. Discover ways of making better use of existing resources.
   Provide powerful tools to augment the expertise of current and future engineers.

The last alternative is adapted for this paper, the typical "engineering" approach to a resource allocation problem: that of "cutting the pie" into smaller slices by more efficient use of the existing resource.

fficulties reprinting the figures,
sitt's article has been reprinted
om the conference proceedings.
with permission - ed.

COMPUTER-BASED LEARNING, EXPERT SYSTEMS AND SOFTWARE ENGINEERING: ADVANCED TOOLS FOR ENGINEERING EDUCATION NOW AND IN 2001.

Dr. Kenneth L. Modesitt
Head, Department of Computer Science
Western Kentucky University
Bowling Green, KY 42101 (USA)
(502) 745-4642

English Abstract

ill design and build space station
om of today and the lunar and Martian
s and bases of tomorrow? Several
ced tools can address the shortage of
fied future engineers. Expert
ter systems permit experts to
mit their expertise to successors.
ter-based learning tools help people
difficult concepts by interactive
gues. Software engineering methods
e large programs to be built with a
r degree of reliability and safety.

German Abstract

Wer wird heute die Raumstation "Freedom" und in der Zukunft die Mond- und Marsstation entwerfen und bauen? Es gibt mehrere fortgeschrittene Möglichkeiten. Expert Computer Systems erlauben Experten, ihre Erfahrungen an Nachfolger weiterzureichen. Intelligente Lernsysteme helfen Studenten, schwierige Konzepte im gegenseitigen Dialog zu erlernen. Techniken in "Engineering Software" machen es möglich, grosse Programme mit einem höheren Grad von Zuverlässigkeit zu entwickeln.

What is the Problem?

he world has never lacked for
ems or "opportunities". Moreover, it
lso been recognized by at least one
rity that problems are not usually
d by the people who create them:
e world that we have made as a result
the level of thinking we have done
s far creates problems we cannot
ve at the same level of thinking at
ch we created them."
lbert Einstein
es not take a person of Albert's
lect to perceive that the world is
te with a multitude of significant
ems of our own making: the greenhouse
t, acid rain, de-forestation,
tion, bloody altercations over
aries of nations, crime, food
ies to needy areas, ad infinitum.
ome reasons for these include: lack
ney, lack of motivation, lack of
n, fear of loosing current resources,
The single most important lack
-wide, however, is that of
ically competent professional
iduals who can attack the gaps
en "what is" and "what should be."
the theme of this conference:
eering education 2000, as held in an
national environment.
s a specific example, figure 1 shows
earth of technical currently-
fied staff at a major aerospace
ration. The figure demonstrates the
h of senior knowledge, promise in be-
ng technical staff, and a gap in the
ctive mid-range. Similar plots would
nationwide for most engineering
ed companies and universities!

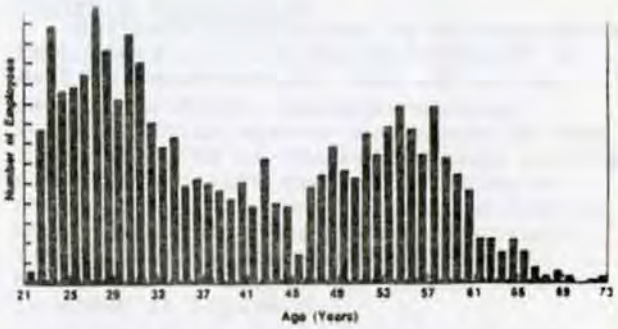


Figure 1. Distribution of Engineers

Solving the Problem

There are a number of approaches to solve the problem of increasing the effectiveness of a resource, i.e., the pool of technically qualified people.

1. Find a new supply of the resource.
   Increase enrollment efforts for women, minorities, third world, under-represented populations.
   Retrain other professionals.
2. Find a new type of resource.
   Use robots to perform tasks now done by clever humans. Not viable!
3. Discover ways of making better use of existing resources.
   Provide powerful tools to augment the expertise of current and future engineers.

The last alternative is adapted for this paper, the typical "engineering" approach to a resource allocation problem: that of "cutting the pie" into smaller slices by more efficient use of the existing resource.

fficulties reprinting the figures,
sitt's article has been reprinted
om the conference proceedings.
with permission - ed.

Proceedings
Frontiers in Education Conference
Ingenieurpädagogik '90

Software Solutions

The virtual explosion of computing hin the lifetimes of most of us is ctically unknown in history. Never has h a powerful tool been adopted so ckly world-wide. See figure 2 for an ication of how dominant computing has ome. In the United States, the amount rapidly approaching that spent in the ire automotive industry.

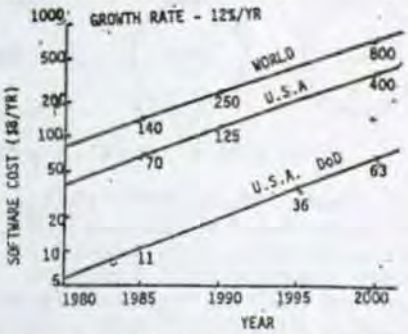
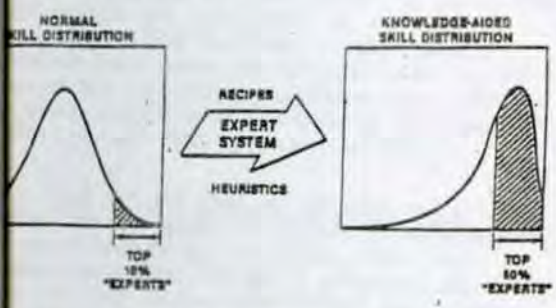


Figure 2. Software Cost Trends

We will look at three major software utions for augmenting human expertise: wledge-based systems, computer-based rning, and software engineering. A rt definition of each will be followed historical perspective and future jections. Finally, we suggest specific s to turn the future into reality.

wledge-based Systems (KBSs)

The rationale for the existence of s may be paraphrased as: "enable a puter to behave more intelligently." se are computer programs which make ensive use of human experiential wledge, and which exhibit more telligence" than is usually displayed normal programs. The subset of KBSs wn as expert systems is able to perform par with human experts in very narrow ains, although very fragile at their daries. This author has worked in the d a number of years, most recently on Space Shuttle Main Engines[1]. A sical reference in KBSs is by the late ld Waterman[2]. See figure 3 for why are attractive.



gure 3. Skill Distribution with KBSs

Computer-based Learning (CBL)

The rationale for the existence of CBL may be paraphrased as: "enable a human being to behave more intelligently." The use of computers in the learning process, both in education and industry, is nearly as old as the computer itself. A recent paper at the 31st annual conference of the leading such professional organization noted that ACM, the professional computing society, was only 10 years older[3]. Use of CBL involves several features: tutor (the student is instructed using drill and practice, dialog simulation, inquiry, or tutorials. This is classical computer-assisted instruction.); tool (the computer has some useful capability programmed into it such as graphics, statistical analysis, integration, word processing, etc.); tutee (the student programs the computer to perform some task).

An example of CBL at its best is the Plato system, now known as NovaNet, with its genesis at the University of Illinois in 1959, now offering millions of hours of delivered courses at thousands of sites in schools and industries worldwide.

Software Engineering

Software engineering is an engineering technology tool for the development of quality, easy-to-use, cost-effective, easy-to-maintain, schedule-meeting software which solves real needs of the customer. Use of the methodology involves the software life cycle consisting of needs assessment, requirements analysis, cost estimating, design, construction, test, installation, and maintenance/ enhancement. One of the modern variants is shown in figure 4.

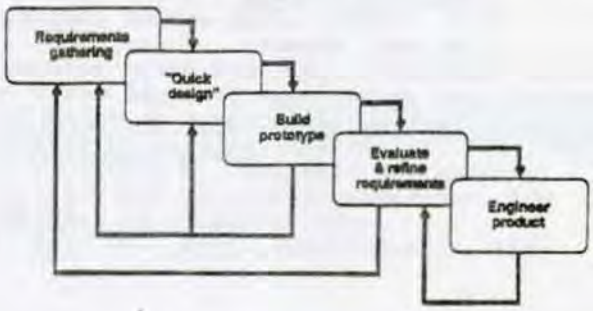


Figure 4. Modern Software Development

This author has been active in the field since the early 1960s[4]. The most common examples of the resulting products are large systems used in the space program, FAA, banking, airlines, military, etc., where program size may exceed 1,000,000 lines of code and development time is measured in tens of person-years. The Bureau of Labor Statistics estimates a demand for over 900,000 such software engineers -- the supply is 600,000.

Past, Present, and Future Software Tool Relationships

t (1970) Twenty years ago, KBS, CBL, software engineering (SWE) had very tle to do with another -- indeed, it difficult to find any collaborative orts. It was as though no one had even rd of the other fields. Figure 5 onstrates this gap.

Figure 5. Past Relationship

sent (1990) Currently, there is siderable awareness that pairs of ds, at least, have something in non. ASEE and IEEE sponsor conferences he relationships of KBSs and software neering. The fields are finally ing closer. See figure 6.

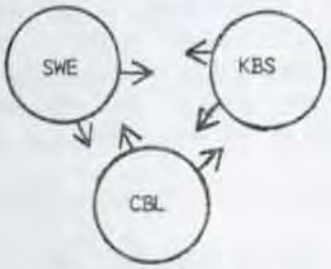
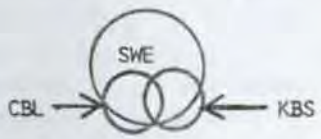


Figure 6. Current Relationship

re (2000) A possible and desirable would be for considerable overlap to t among all three fields (Figure 7).



igure 7. Possible Future Relationship

ach subfield still has some unique on, but substantial parts of both CBL BS would be included in software eering. KBS methods would permit types and domains of problems to be ssed by software engineering. CBL iques would add immensely to the of-use of the software product.

How Do We Get to the Future?

he future goal of strong cooperation professionals in KBSs, CBL and are engineering will provide an r to the issue addressed earlier: "Provide powerful tools to augment the expertise of current and future engineers."

Future engineers will be addressed via education: class projects emphasizing teams, real problems, use of best tools, evaluation, use at beginning of academic program, cooperation with industry.

Current engineers will be aided by training with: teams, real problems, use of best tools, evaluation, cooperation with universities.

Solutions to the problems listed at the beginning of this paper are much more likely with qualified technical professionals who result from such programs. However, current problems are not the only issue with which humankind must be concerned. A recent speculation is that the main technology drivers in the next century will be space, computers, energy, and communication[5]. Another way to address the shortage of engineering talent in one country is to cooperate and combine efforts and resources of several nations as is now done in space.

Results

Use of the suggestions above will diminish the number and severity of many significant problems confronting us today. Increased use of the software tools of computer-based learning, knowledge-based systems and software engineering will be a positive impact for engineering education now and in 2001.

References

1. Modesitt, K., Inductive Knowledge Acquisition Experience with Commercial Tools for Space Shuttle Main Engine Testing, Fifth Conference on Artificial Intelligence for Space Application, NASA/ University of Huntsville, AL, 1990.
2. Waterman, D., A Guide to Expert Systems, McGraw-Hill, 1986.
3. Modesitt, K., Lessons Learned in Computer-based Learning: A Personal Tale of Three Decades. Association for the Development of Computer-based Instructional Systems Conference, Washington, D.C.,1989.
4. Modesitt, K., Software Engineering Training for TI Middle Management. ACM National Conference, 1980.
5. Piel, G., 2081, Prentice-Hall, 1981.

Biographical Sketch

Dr. Kenneth L. Modesitt heads the Computer Science Department at Western Kentucky University and teaches courses in all the topics covered in this paper. He has worked for Control Data, IBM, Texas Instruments and for Rockwell, where he helped develop an expert system. A member of the IEEE Computer Society, Modesitt's background in artificial intelligence (AI) begins with Stanford in 1963. His other degrees are from University of Illinois, Carnegie and Washington State University.

CS240 Lab

This year Mrs. Wilson and Dr. Pigford have taught experimental sections of CS240L. Students were given the choice of taking CS240 with or without a one credit lab. The lab is similar to a lab in other science courses. During the two hour lab session the student works on a specific project under the supervision of a faculty member. Each lab assignment concentrates on a specific topic such as "Looping Structures" or "Two Dimensional Arrays". The feedback from the students has been very positive. They feel that their foundation in Pascal is stronger because they spent productive "time on task" under the direction of a teacher.

$1.00 for a PC

The UNIX Lab is raffling a micro-computer to earn money for a badly needed modem. The machine is a Hewlett-Packard 8088 with a 15 Meg hard drive, non-standard 3.5" floppy drive, MS-DOS 2.0, touchscreen monochrome monitor, serial and parallel ports, etc. Documentation, cables, etc comes with it. Purchase several tickets in the UNIX Lab.

IMPORTANT NUMBERS

TCCW Lab	2541
Grise Lab	2705
TCP (CEB) Lab	2970
ACRS	4981
Machine Room (STH 110)	4982
Computer Science Office	4642
Domino's Pizza	781-9494

Automated Reasoning Course

This summer the topics course CS475/G will be covering an introduction to automated reasoning. Dr. Shindhelm will introduce students to this subfield of AI which involves using a program as a tool in "reasoning." The two theorem provers on the VAX, ITP and OTTER will be used extensively. Prerequisite is CS442 or permission of the instructor.

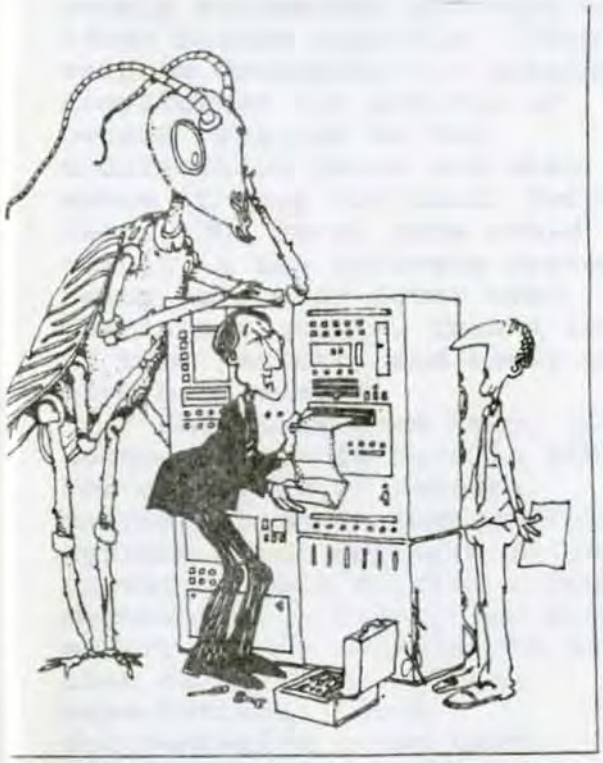
LAB HOURS

For late night programmers, (we don't do programs the night before they're due, do we?) TCCW lab is now open until 2 a.m. Sunday through Thursday. For more information on lab hours, type HELP HOURS on the VAX.

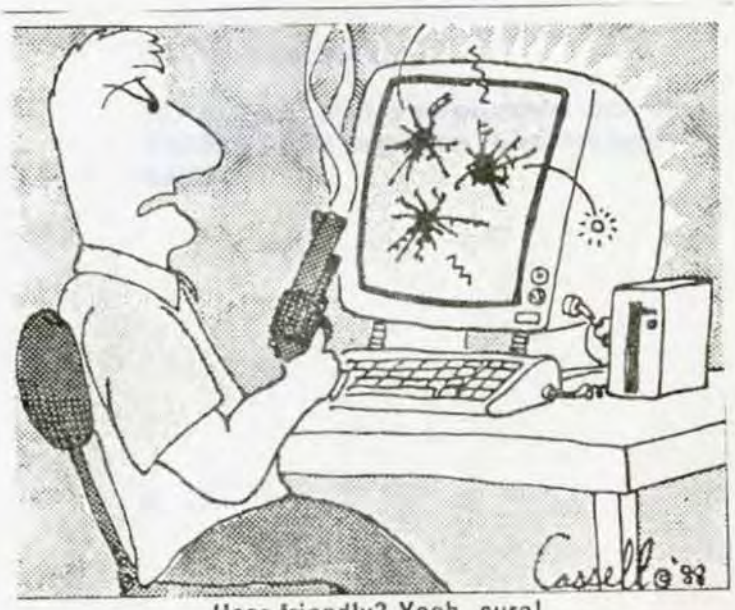
The Mac Project

ACM has finally found a use for the money we had tied up in grants and in our savings account. We have put in an order for a Mac SE with an integrated software package. This machine will be available to ACM members on a demand basis. Plans for the future include buying a laser printer to go with the Mac, expanding our software base, and setting up a Mac Local Area Network (in the far, far future). The SE that was in the department office is now in the UNIX lab. Feel free to use it.

The Weekly Smile



"... Haven't found any bugs yet!"



User-friendly? Yeah, sure!

Dennis the Menace

"NOW MARGARET'S REALLY GONNA BE A KNOW-IT-ALL. HER DAD JUST GOT A FACTS MACHINE!"

"But it was so friendly in the store!"

Baur (concluded)

should require a minimal time in the modification phase, except for changes requested after the original formalization stage.

The final step in the development phase is the preparation of documentation for the software system. This step is very important since a poorly documented software will often become unusable. Poorly written documentation greatly complicates the efforts of people assigned to the modification phase and also makes it very difficult for the user. The worst case could result in the software system being seldom or never used. If the latter occurs, then a lot of time, effort, and money will have been wasted.

Just like good help, good documentation is hard to find. For a variety of reasons, documentation on most software systems rates mediocre at best. Providing well-written system documentation takes time and effort and is usually the area that falls short of user expectations. Good documentation makes good software better; poor documentation can make good software unusable.

Sloan (concluded)

experiences in actual costs of installation and maintenance, reliability, and flexibility in expansion.

Beckley (concluded)

PC users and resources on the STARMASTER system and other LANs.